二分、倍增

2019年2月4日 13:13

一. 整数集合上的二分

1. 单增序列中找>=x的最小数（即x或x的后继）

i. while(l<r){

```
int mid= l+r >>1;
if(a[mid]>=x)    r=mid;  //mid满足条件
else     l=mid+1;        //mid不满足条件
} return a[l];
```

ii. mid取不到r，因此r初值可取最大值+1，若最后答案为最大值+1，说明无答案

2. 单增序列中找<=x的最大数（即x或x的前驱）

i. while(l<r){

```
int mid= l+r+1 >>1;
if(a[mid]<=x)    l=mid;  //mid满足条件
else     r=mid-1;        //mid不满足条件
} return a[l];
```

ii. r=l+1时，假设取mid= l+r >>1。若mid满足，则死循；若mid不满足，则r取l-1，不符题意，因此r取mid-1对应mid=(l+r+1)>>1

iii. mid取不到l，因此l初值可取最小值-1，若最后答案为最小值-1，说明无答案

3. 右移向下取整，适用于负数二分，而整除二向零取整

二. 实数域上的二分

1. 按某个精度计算

i. while(l+1e-5<r){

```
double mid=(l+r)/2;
if(calc(mid)      r=mid;  //mid满足条件
else     l=mid;} //mid不满足条件
```

2. 循环固定次数

i. for_(i,0,100){

```
double mid=(l+r)/2;
if(calc(mid)      r=mid;  //mid满足条件
else     l=mid;} //mid不满足条件
```

三. 三分求单峰凹/凸函数极值

1. 对于一半严格单增一半严格单降的函数，可以三分求极大值

```
double three_devide(double low,double up) {
    double m1,m2;
    while(up-low>=eps)   {
        m1= low+ (up-low)/3;
        m2= up- (up-low)/3;
        if(f(m1)<=f(m2))//f为计算函数
            low=m1;
        else
            up=m2;   }
    return (m1+m2)/2;    }
```

2. 极小值

```
ld l=0,ml,mr,r=1e9;
for(int i=0;i<100;++i){
    ml=l+(r-l)/3, mr=r-(r-l)/3;
    if(f(ml)<=f(mr)) r=mr;
    else l=ml;}
```

四. 倍增解决RMQ（Sparse table算法）

1. 令f[i][j]表示[i,i+ 2^j -1]区间内的最值，显然f[i][0]=a[i]
2. 倍增递推时，f[i][j] = f[i][j-1] 或 f[i+(1<< j-1)][j-1]（正右一半J，往右一半J位置的正右一半J）
3. 注意这个递推过程的阶段是j，要放在最外层
4. 长度为t的[l,r]的最值是f[l][lg[t]]或f[r-（1<<lg[t]）+1][lg[t]]，即向下取到2的整数倍后，取左半和右半的最值，为了覆盖到r，要让r先-去整数倍，再+1
5. 为了让常数更少，可以在预处理前先把log2(i)存到各个lg[i]里

```
void init(){
    for__(i,1,n)
        st[i][0] = a[i];
    for__(j,1,lg[n])
        for__(i,1,n-(1<<j)+1)
            st[i][j] = max(st[i][j-1], st[i+(1<<j-1)][j-1]);}
inline int ask(int l,int r){
    int x=lg[r-l+1];
    return max(st[l][x], st[r+1-(1<<x)][x]);}
```

6. 例题：动态在队尾插入新数据，询问从队尾往前看的j个数中的最大值
   i. 只需反向存储区间最值即可
   ii. 每次加入一个数，都对新位置做一次初始化
   iii.
```
void add(int x){
    a[++r] =x;
    st[r][0] =x;
    int ed= lg[r];
    for__(j,1,ed)
        st[r][j]= max(st[r][j-1],st[r- (1<< j-1)][j-1]);}
```
   iv.
```
int ask(int l,int r){
    int x= log2(r-l+1);
    return max(st[r][x],st[l -1 +(1<<x)][x]);}
```

-
- 例题

五. 剪网线

1. 题意：n根网线能切成的k根尽量长的等长网线（不够长的扔掉）
2.
```
sort(a,a+n,greater<int>());        //先试试排个降序
l=0,r=a[0];                        //初值分别赋为最小和最大
do{
    mid=(l+r+1)>>1;//避免r取l+1时出现mid一直取l的情况
    cnt=a[0]/mid;
    for_(i,1,n){
        cnt+=a[i]/mid;  //利用了一下整除
        if(cnt>k)
            break;}
    if(cnt<k)          //此时mid不符条件，可以缩小上界为mid-1
```

```
            r=mid-1;
        else                //此时mid不一定是最大值，继续二分
            l=mid;
    }while(r>l);
```

六. 切巧克力

1. 题意：同上，输出去尾法保留到一位小数
2. 注意：因为是去尾法，为了精度，最后应输出r
3. sort(d,d+n,greater<double>()); //降序

```
double l= 0;
double r= d[0];
while(r-l > 1e-6){
    double mid= (l+r)/2;
    cnt= d[0]/mid;
    for_(i,1,n){
        cnt+= d[i]/mid;
        if(cnt>=k)
            break;}
    if(cnt>=k)
        l= mid; //尝试找更大值
    else
        r= mid; //尝试找满足值}
printf("%d.%d",int(r*10)/10,int(r*10)%10);
```